a program memory comprising I individually addressable, parallel-connected memory banks with I being at least equal to N, said program memory comprising a program recorded in an interlaced fashion at a rate of one code per memory bank and per address applied to said memory banks; and

reading means for reading said program memory by reading a code in each of said I memory banks during a cycle for reading an instruction, with a cycle comprising at least one code to be read, and when a number of codes of the instruction read is less than I, comprises codes belonging to a following instruction.

21. A signal processor according to Claim 20, wherein said reading means comprises address means for applying to said memory banks individual addresses generated from a collective value of a program counter that is incremented, before a beginning of the cycle for reading the instruction, by a value equal to a number of codes comprising a previous instruction.

22. A signal processor according to Claim 21, wherein said address means applies to each of said memory banks an individual read address equal to P0 or P0+1, with P0 being a quotient of a division by I of a value of the program counter.

23. A signal processor according to Claim 22, wherein said address means comprises applying, to an i ranking memory bank, an address equal to P0 when i is greater than R and an address equal to P0+1 when i is less than or equal to R, with R being a remainder of the division by I of the value

2

of the program counter.

24. A signal processor according to Claim 20,
wherein said reading means comprises reorganization means for
reorganizing codes of a sequence of codes read in said program
memory according to an algorithm defined as follows:

$c'(j) = c(i)$, and $i = (j+R')$ modulo I,

with i and j designating ranks of the codes before
and after reorganization, $c(i)$ designating i ranking codes in
their arrangement after being read in said program memory,
$c'(j)$ designating j ranking codes after reorganization, and R'
is a remainder of a division by I of a value that was shown by
the program counter during a previous clock cycle.

25. A signal processor according to Claim 24,
wherein said reorganization means applies to the codes of the
sequence of codes read a circular permutation comprising a
number of circular permutations equal to R' or to I-R',
depending on a direction of the circular permutation made.

26. A signal processor according to Claim 25,
wherein said reorganization means comprises a barrel shifter
having a control input for receiving the parameter R'.

27. A signal processor according to Claim 20,
wherein said reading means comprises filtering means for
filtering codes that do not belong to the instruction to be
read, using parallelism bits accompanying the codes.

28. A signal processor according to Claim 27,
wherein the filtered codes are replaced by no-operation codes.

3

29. A signal processor according to Claim 28,
wherein said filtering means executes an algorithm defined as
follows:

> For $j = 0$,
> $val(j=0) = v$,
> $s(j=0) = c'(j=0)$;
> For j going from 1 to I,
> $val(j) = v$ if:
> $val(j-1) = v$ and if parallelism bit of $c'(j) = p$,
> else $val(j-1) = /v$;
> $s(j) = c'(j)$ if $val(j) = v$;
> $s(j) = NOP$ if $val(j) = /v$,

with $val(j)$ being a validation term associated with
each j ranking code, $c'(j)$ is capable of having two values v
and $/v$, $s(j)$ designates j ranking outputs of said filtering
means corresponding to same ranking inputs receiving a code
$c'(j)$, and NOP indicates a no-operation code.

30. A signal processor according to Claim 29,
wherein said reading means comprises at least one parallel-
connected RISC type execution unit for receiving non-filtered
codes.

31. A processor for executing variable-sized
instructions, each instruction comprising up to N codes, the
processor comprising:
a memory comprising I individually addressable,
parallel-connected memory banks with I being at least equal to
N, said memory comprising a program recorded in an interlaced
fashion; and
a reading circuit for reading said memory by reading

4

a code in each of said I memory banks during a cycle for reading an instruction, with a cycle comprising at least one code to be read, and when a number of codes of the instruction read is less than I, comprises codes belonging to a following instruction.

32. A processor according to Claim 31, wherein the program is recorded at a rate of one code per memory bank and per address applied to said memory banks.

33. A processor according to Claim 31, wherein said reading circuit comprises an address circuit for applying to said memory banks individual addresses generated from a collective value of a program counter that is incremented, before a beginning of the cycle for reading the instruction, by a value equal to a number of codes comprising a previous instruction.

34. A processor according to Claim 33, wherein said address circuit applies to each of said memory banks an individual read address equal to P0 or P0+1, with P0 being a quotient of a division by I of a value of the program counter.

35. A processor according to Claim 34, wherein said address circuit comprises applying, to an i ranking memory bank, an address equal to P0 when i is greater than R and an address equal to P0+1 when i is less than or equal to R, with R being a remainder of the division by I of the value of the program counter.

36. A processor according to Claim 31, wherein said

reading circuit comprises a reorganization circuit for
reorganizing codes of a sequence of codes read in said memory
according to an algorithm defined as follows:

$c'(j) = c(i)$, and $i = (j+R')$modulo I,

with i and j designating ranks of the codes before
and after reorganization, $c(i)$ designating i ranking codes in
their arrangement after being read in said memory, $c'(j)$
designating j ranking codes after reorganization, and R' is a
remainder of a division by I of a value that was shown by the
program counter during a previous clock cycle.

37. A processor according to Claim 31, wherein said
reading circuit comprises a filtering circuit for filtering
codes that do not belong to the instruction to be read, using
parallelism bits accompanying the codes.

38. A processor according to Claim 37, wherein the
filtered codes are replaced by no-operation codes.

39. A processor according to Claim 38, wherein said
filtering circuit executes an algorithm defined as follows:

For j = 0,
val(j=0) = v,
s(j=0) = c'(j=0);
For j going from 1 to I,
val(j) = v if:
val(j-1) = v and if parallelism bit of c'(j)= p,
else val(j-1) = /v;
s(j) = c'(j) if val(j) = v;
s(j) = NOP if val(j) = /v,
with val(j) being a validation term associated with

6

each j ranking code, c'(j) is capable of having two values v
and /v, s(j) designates j ranking outputs of said filtering
means corresponding to same ranking inputs receiving a code
c'(j), and NOP indicates a no-operation code.

40. A processor according to Claim 39, wherein said
reading means comprises at least one parallel-connected RISC
type execution unit for receiving non-filtered codes.

41. A method for reading variable-sized instructions
in a signal processor, with each instruction comprising up to
N codes, the method comprising:
    providing a program memory comprising I individually
addressable, parallel-connected memory banks with I being at
least equal to N;
    recording codes of a program in the program memory
in an interlaced fashion at a rate of one code per bank and
per address applied to the memory banks; and
    during a read cycle of an instruction, reading a
sequence of codes in the I memory banks, the sequence
comprising at least one code of the instruction to be read
and, when a number of codes of the instruction read is less
than I, comprises codes belonging to a following instruction.

42. A method according to Claim 41, further
comprising applying, to the memory banks, individual addresses
generated from a collective value of a program counter that is
incremented, before a beginning of the read cycle for the
instruction, by a value equal to a number of codes contained
in a previous instruction.

43. A method according to Claim 42, further comprising applying, to each to the memory banks, an individual read address equal to P0 or P0+1, with P0 being a quotient of a division by I of the value of the program counter.

44. A method according to Claim 43, further comprising applying, to an i ranking memory bank, an address equal to P0 when i is greater than R and an address equal to P0+1 when i is less than or equal to R, with R being a remainder of the division by I of the value of the program counter.

45. A method according to Claim 41, further comprising reorganizing codes of the sequence of codes read in the program memory according to an algorithm defined as follows:

$c'(j) = c(i)$, and $i = (j+R')$ modulo I,

with i and j designating ranks of the codes before and after reorganization, $c(i)$ designating i ranking codes in their arrangement after reading in the program memory, $c'(j)$ designates j ranking codes after reorganization, and with R' being a remainder of a division by I of a value that was shown by the program counter during a previous clock cycle.

46. A method according to Claim 41, further comprising filtering codes read that do not belong to the instruction to be read, using parallelism bits accompanying the codes.

47. A method according to Claim 46, wherein the

8

filtered codes are replaced by no-operation codes.

48. A method according to Claim 47, wherein the
codes are filtered according to an algorithm defined as
follows:

For $j = 0$,

$val(j=0) = v$,

$s(j=0) = c'(j=0)$;

For j going from 1 to I,

$val(j) = v$ if:

$val(j-1) = v$ and if parallelism bit of $c'(j) = p$,

else $val(j-1) = /v$;

$s(j) = c'(j)$ if $val(j) = v$;

$s(j) = NOP$ if $val(j) = /v$,

with $val(j)$ being a validation term associated with
each j ranking code, $c'(j)$ is capable of having two values v
and $/v$, $s(j)$ designates j ranking outputs of the filtering
corresponding to same ranking inputs receiving a code $c'(j)$,
and NOP indicates a no-operation code.

49. A method according to Claim 48, wherein non-
filtered codes are sent to parallel-connected RISC type
execution units.

50. A method for reading variable-sized instructions
in a processor, with each instruction comprising up to N
codes, the signal processor comprising a memory comprising I
individually addressable, parallel-connected memory banks,
with I being at least equal to N, the method comprising:

recording codes of a program in the memory in an
interlaced fashion; and

9

_____/

during a read cycle of an instruction, reading a sequence of codes in the I memory banks, the sequence comprising at least one code of the instruction to be read and, when a number of codes of the instruction read is less than I, comprises codes belonging to a following instruction.

51. A method according to Claim 50, wherein the program is recorded at a rate of one code per bank and per address applied to the memory banks

52. A method according to Claim 50, further comprising applying, to the memory banks, individual addresses generated from a collective value of a program counter that is incremented, before a beginning of the read cycle for the instruction, by a value equal to a number of codes contained in a previous instruction.

53. A method according to Claim 52, further comprising applying, to each to the memory banks, an individual read address equal to P0 or P0+1, with P0 being a quotient of a division by I of the value of the program counter.

54. A method according to Claim 53, further comprising applying, to an i ranking memory bank, an address equal to P0 when i is greater than R and an address equal to P0+1 when i is less than or equal to R, with R being a remainder of the division by I of the value of the program counter.

55. A method according to Claim 50, further

10

comprising reorganizing codes of the sequence of codes read in the memory according to an algorithm defined as follows:

$c'(j) = c(i)$, and $i = (j+R')$ modulo I,

with i and j designating ranks of the codes before and after reorganization, $c(i)$ designating i ranking codes in their arrangement after reading in the program memory, $c'(j)$ designates j ranking codes after reorganization, and with R' being a remainder of a division by I of a value that was shown by the program counter during a previous clock cycle.

56. A method according to Claim 50, further comprising filtering codes read that do not belong to the instruction to be read, using parallelism bits accompanying the codes.

57. A method according to Claim 56, wherein the filtered codes are replaced by no-operation codes.

58. A method according to Claim 57, wherein the codes are filtered according to an algorithm defined as follows:

For $j = 0$,
$val(j=0) = v$,
$s(j=0) = c'(j=0)$;
For j going from 1 to I,
$val(j) = v$ if:
$val(j-1) = v$ and if parallelism bit of $c'(j) = p$,
else $val(j-1) = /v$;
$s(j) = c'(j)$ if $val(j) = v$;
$s(j) = NOP$ if $val(j) = /v$,
with $val(j)$ being a validation term associated with

11